

REMARKS

Claims 1-5, 7-18, and 20-28 are pending in the present application. Claims 6 and 19 were canceled. Reconsideration of the claims is respectfully requested.

In addition, the Office Action Summary indicated that the drawings filed on June 10, 1999 were objected to by the examiner. However, no mention of the objection was mentioned in the detailed action section of the Office Action. In a telephone conversation initiated by the applicant on December 22, 2003, the examiner instructed to disregard the objection at this time.

I. 35 U.S.C. § 103, Obviousness

The examiner has rejected claims 1-3, 6, 11, 13-16, 19, 24, and 26-28 under 35 U.S.C. § 103(a) as being unpatentable over *Stiles* (US 6,314,446 B1) in view of *Hashun* (US 5,640,529). This rejection is respectfully traversed.

With regard to claims 1, 11, 14, 24, 27, and 28, the examiner states:

Referring to claim 1, 11, 14, 24, 27, and 28, *Stiles* discloses a data processing system for monitoring a plurality of related threads with the following steps:

- polling the plurality of related threads for status information ("*polls the monitored threads and second, it allows the updating of the display, that is the created window for status.*", col. 3, lines 44-67);
- responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive ("*The method displays the resulting status to the user in an intuitive manner with four configurations indicating: process running normally, process running intermittently, process stopped normally, and process halted unexpectedly.*" see Abstract);

Stiles fails to explicitly teach:

- responsive to a determination that a thread within the plurality of related threads is inactive, initiating cleanup processes for the thread based on the status information. However, *Hashun* teaches that when threads are in a period of inactive status, a clean-up is performed ("*Performing clean-up during these inactive periods is usually adequate to maintain sufficient free memory for the needs of host central processing unit 52.*", col. 3, lines 35-45). It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of cleanup during inactive periods for the reason of increasing efficiency ("*Execution of clean-up states during the periods of time when host CPU 52 is inactive is generally adequate to maintain sufficient free memory because CPU 52 generally reads and writes in bursts.*", col. 8, lines 54-57).

(Office Action, dated September 3, 2003, pages 3-4).

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). Amended independent claims 1, 11, 14, 24, 27, and 28 recite the features of polling a thread for status information, determining whether a thread is inactive in response to receiving the status information, and responsive to a determination that the thread is inactive, initiating cleanup processes for the thread based on the status information. Amended independent claim 1, which is representative of amended independent claims 11, 14, 24, 27, and 28, is reproduced below:

1. A method in a data processing system for monitoring a plurality of related threads, the method comprising the data processing system implemented steps of:
polling the plurality of related threads for status information;
responsive to receiving the status information, determining whether a thread within a plurality of related threads is inactive; and
responsive to a determination that a thread within the plurality of related threads is inactive, initiating cleanup processes for the thread based on the status information.

The claimed invention teaches the feature of initiating a clean up process for a thread based on the status information. Applicants agree with the examiner that *Stiles* does not teach this feature.

However, *Hasbun* does not cure the deficiencies of *Stiles*. Although *Hasbun* performs a cleanup process using a state machine, *Hasbun* does not teach or suggest performing a cleanup process for a thread based a determination that the thread is inactive.

The examiner states that *Hasbun* teaches this feature in the following passage:

Performing clean-up during these inactive periods is usually adequate to maintain sufficient free memory for the needs of host central processing unit 52. However, reclamation of dirty memory during periods of host inactivity can be overrun if host CPU 52 writes continuously to solid state disk 60. Once reclamation of dirty memory is overrun solid state disk 60 may not be able to perform host writes in the period of time allotted by industry standard interface specifications. When this occurs a write error could be displayed to the computer user on monitor 54. (*Hasbun*, col. 3, ll. 35-45).

Although the passage above mentions that a cleanup process is performed when a the host CPU is in a period of inactivity, there is no mention in *Hasbun* of performing the cleanup process of a thread based on determining that the thread is inactive. Rather, *Hasbun* teaches that the cleanup

process is performed when a command interrupt is issued from the CPU. For example, the cleanup process of the solid state disk in *Hasbun* is described in the following section:

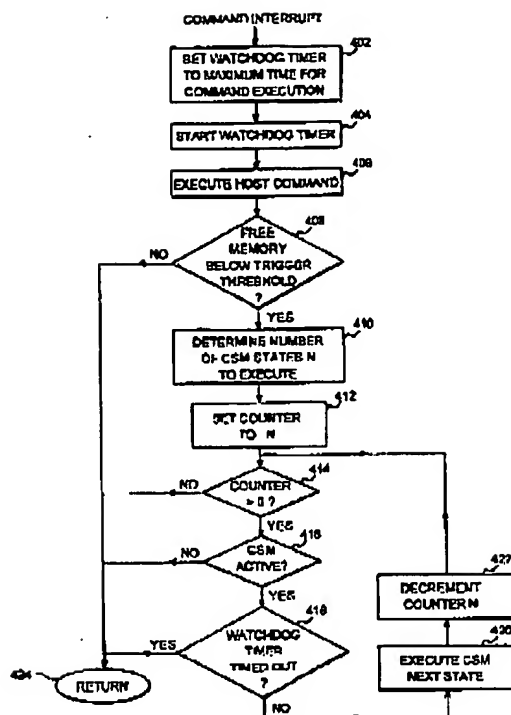
Solid state disk drive 60 achieves write speeds close to conventional magnetic disk drives by writing a sector of data to a new location each time it is revised, rather than erasing the previous location and writing the revised data to that same physical location. As a result of this practice, solid state disk 60 becomes sprinkled with dirty sectors, that is to say invalid user data. Recovering the memory space occupied by dirty sectors mandates clean-up. Stated slightly differently, the write practices of solid state disk controller 64 require that invalid user data be converted into free memory.

Execution of clean-up states during the periods of time when host CPU 52 is inactive is generally adequate to maintain sufficient free memory because CPU 52 generally reads and writes in bursts. As a result, there are relatively long periods of time when microprocessor 92 is free to perform clean-up. **Execution of clean-up states during host command execution prevents clean-up from being overrun during prolonged host writes to FLASH array 62.**

Clean-up is managed by a finite state machine, called a clean-up state machine (CSM). That is, clean-up is achieved using a finite number of states, or services, which are chained together. Each clean-up state points to the next state, in effect chaining each state to another state. Each state takes no more than 500 .mu.seconds of CPU 92 time to execute.

Clean-up may use a single clean-up state machine or two clean-up state machines, and is granted execution time when host interface 152 is inactive. **Clean-up is initiated when an enable CSM service is called. The enable CSM service is called whenever the host writes to solid state memory disk 60.** This service activates a clean-up state machine by setting a CSM next state pointer to a clean-up state. (*Hasbun*, col. 8, line 43 to col. 9, line 7).

In addition, Figure 8 of *Hasbun* illustrates the method of performing a cleanup during time allotted to host command execution:



The passage and figure above show that *Hasbun* teaches converting invalid user data to free memory within a solid state memory disk during the time allotted to execute commands from a host CPU. The CPU issues a command interrupt to the state machine controller, which executes the command. The cleanup is initiated when an enable client state machine is called. The enable client state machine is called whenever the host writes to a solid state memory disk. Thus, *Hasbun* teaches that a cleanup process is performed as a result of the host CPU write to a solid state memory disk. There is no teaching or suggestion in *Hasbun* to initiate cleanup of a thread's resources simply in response to a determination that the thread is inactive as in the claimed invention.

Furthermore, *Hasbun* does not teach the problem or its source. The present invention recognizes the problem of determining whether a thread within a multi-threaded process is inactive, and if so, cleaning up that inactive thread's resources to enable a more efficient use of the system's resources. *Hasbun* does not teach the problem or its source. Instead, *Hasbun* is directed towards a system for converting invalid user data within a solid state disk into free memory during time allotted to execute a host command from a standard interface. *Hasbun* teaches a system for preventing conversion of invalid user data to free memory from being

overrun during prolonged user reads or writes to a solid state disk. One of ordinary skill in the art would therefore not be motivated to modify the reference in the manner required to form the solution disclosed in the claimed invention.

Moreover, stating that it is obvious to try or make a modification or combination without a suggestion in the prior art is not *prima facie* obviousness. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also see *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Examiner may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification. In making an obviousness determination, one cannot pick and choose among the individual elements or assorted prior art references to recreate the claimed invention. *Symbol Technologies, Inc. v. Opticon, Inc.*, 935 F.2d 1569, 19 U.S.P.Q.2d 1241 (Fed. Cir. 1991). Instead, whether the prior art made obvious the invention must be determined by looking for some teaching or suggestion in the references to support their use in the particular claimed invention. *Id.* The claimed invention includes the feature of performing a cleanup process in response to a determination that a thread within the plurality of threads is inactive. In contrast, *Stiles* teaches displaying the status of tasks, and *Hasbun* teaches performing a cleanup process of a solid state disk drive. However, neither reference teaches or suggests initiating a cleanup process for an inactive thread. The rationale given for the modification is not based on the prior art. Instead, the examiner states his personal opinion using hindsight and "increased efficiency" as the rationale. The fact that a thread is inactive does not mean that one would perform a cleanup process on it. For example, the thread may be alive but suspended and inactive. One would not necessarily desire to perform a cleanup process on that inactive thread. Thus, since neither *Stiles* nor *Hasbun* teach performing the cleanup process in response to a determination that a thread is inactive, nor does the examiner point to any reference that suggests this feature be combined with the contents of the *Stiles* and *Hasbun* references, applicants respectfully submit that *Stiles* and *Hasbun* cannot be modified to produce the claimed invention.

Furthermore, neither *Stiles* nor *Hasbun* teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. In fact, *Hasbun* actually teaches away

from the presently claimed invention because it performs its cleanup process based on the receipt of a command interrupt from the host CPU, rather than performing its cleanup process in response to a determination that the thread is inactive. Absent the Examiner pointing out some teaching or incentive to implement *Stiles* and *Hasbun* to perform a cleanup process on a thread in response to on a determination that a thread is inactive, one of ordinary skill in the art would not be motivated from the references to make the changes necessary to derive the present invention from the reference's teachings.

Therefore, the rejection of claims 1, 11, 14, 24, 27, and 28 under 35 U.S.C. § 103 has been overcome.

II. Dependent Claims

If an independent claim is nonobvious under 35 U.S.C. §103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Claims 2-5, 7-10, 12-13, 15-18, 20-23, and 25-26 are dependent claims that depend on independent claims 1, 11, 14, and 24. Applicants have already demonstrated claims 1, 11, 14, and 24 to be in condition for allowance. Applicants respectfully submit that claims 2-5, 7-10, 12-13, 15-18, 20-23, and 25-26 are also allowable, at least by virtue of their dependency on allowable claims.

Therefore, the rejection of dependent claims 2-5, 7-10, 12-13, 15-18, 20-23, and 25-26 under 35 U.S.C. § 103 has been overcome.

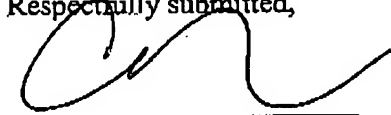
III. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 12/24/03

Respectfully submitted,



Cathrine K. Kinslow
Reg. No. 51,886
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants